Volume 4, Issue 9, September 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Verilog Implementation of UART with Status Register

B. Silpa
PG Student,Department of ECE,
S.V.C.E.T., Chittoor,
A.P., India

K. Lokesh Krishna
Associate Professor, Department of ECE.
S.V.C.E.T., Chittoor,
A.P., India

K. Anuradha
Professor, Department of CSE
L.B.C.E.W., Visakhapatnam,
A.P., India

Abstract— In parallel communication the cost as well as complexity of the system increases due to simultaneous transmission of data bits on multiple wires. Serial communication alleviates this drawback and emerges as a effective candidate in many applications for long distance communication as it reduces the signal distortion because of its simple structure. This paper focuses on the Verilog HDL implementation of UART with status register which supports asynchronous serial communication. This paper presents the architecture of UART which indicates, during reception of data, parity error, framing error, overrun error and break error using status register. In the proposed method for error correction hamming code is used. By using the hamming code, single bit error can be detected and corrected. The whole design is functionally verified using Xilinx ISE Simulator. In this paper we propose a technique for software implementation of an UART with the goal of getting a customizable UART-core which can be used as a module in implementing a bigger system irrespective of one's choice of implementation platform. The simulation results as well as the test results are seen to be satisfactory.

Keywords— Universal Asynchronous Receiver Transmitter; status register; Verilog implementation; Encoder, Decoder and asynchronous serial communication.

I. INTRODUCTION

In parallel communication the cost as well as complexity of the system increases due to simultaneous transmission of data bits on multiple wires. Serial communication alleviates this drawback and emerges as effective candidate in many applications for long distance communication as it reduces the signal distortion because of its simple structure. Universal Asynchronous Receiver Transmitter (UART) is a kind of serial communication protocol to support full duplex communication. It is mostly used for short- distance, low speed, low-cost data exchange between computer and peripherals. UARTs are used for asynchronous serial data communication by converting data from parallel to serial at transmitter with some extra overhead bits using shift register and vice versa at receiver. It is generally connected between a processor and a peripheral. To the processor the UART appears as an 8-bit read/write parallel port. [1] - [6]. It has many advantages such as simple resources, reliable performance, strong anti- jamming capability and easy to operate.

The UART is a large scale integrated circuit which contains all the software programming necessary to fully control the serial port of a PC (Personnel computer). UART performs parallel-to-serial conversion on data bits received from the host processor into serial data stream, and serial-to-parallel conversion on serial data bits received from serial device to the host processor**.** It also adds the start and stop bit to the data for synchronization. In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media and to regulate the flow of data in the event that the remote device is not prepared to accept more data [7]-[10].

In general the UART consists of three sub-modules namely baud rate generator, receiver and transmitter. The baud rate generator is used to produce a local clock signal which is much higher than the baud rate to control the UART to receive and transmit, the UART receiver module is used to receive the serial signals at RXD, and convert them into parallel data and the UART transmit module converts the parallel signal into serial bits according to the basic frame format and transmits those bits through TXD [11].

The outline of the work is as follows. Section II discusses about the proposed UART architecture. Section III presents the simulation results of the proposed UART and Section IV presents conclusion.

II. PROPOSED UART ARCHITECTURE

The proposed architecture of UART is shown in the figure 1.The proposed architecture consists of transmitter, receiver, baud rate generator and line control register. Asynchronous communication is done using UART wherein the clock information is not communal between the two units that are communicating with each other. For synchronization some overhead bit are added to data bits while transmission which specifies that the data bits are transmitted in the form of frame. The process of de- framing is done at the receiver input once the frame is received.

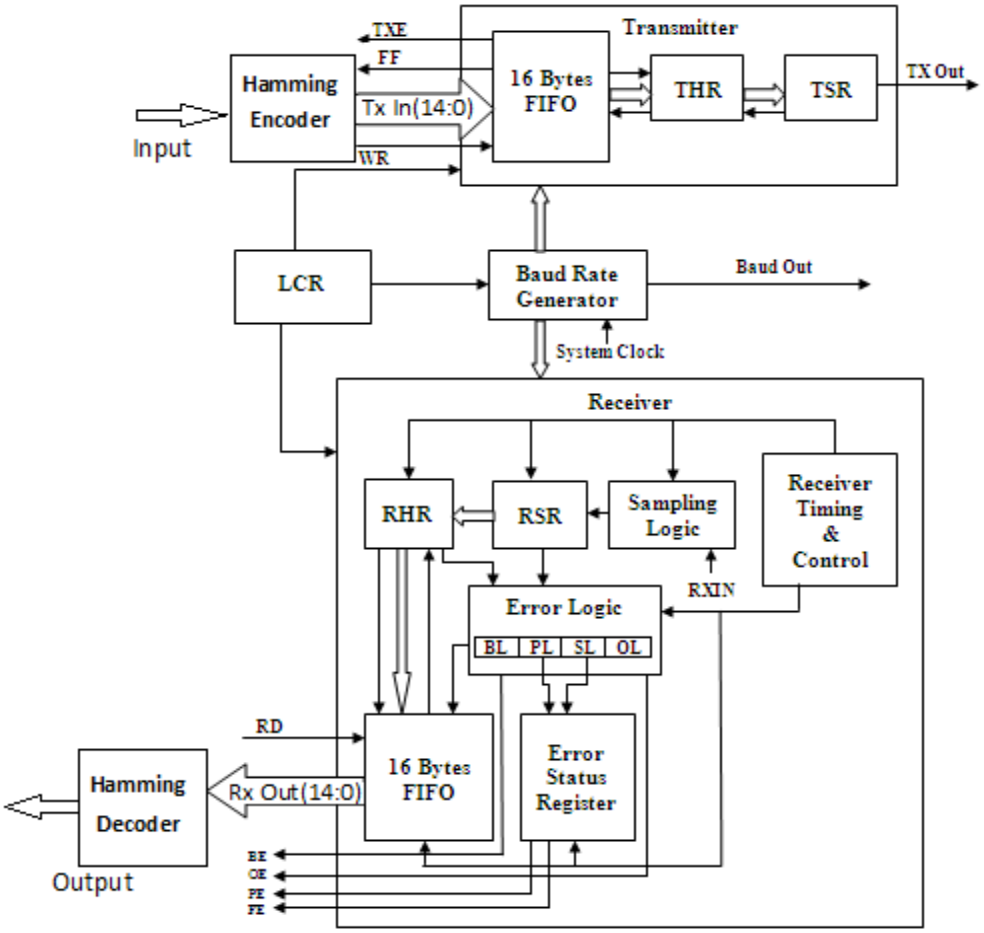

Fig.1. Proposed architecture of UART

A. Line Control Register (LCR):

The line control register (LCR) is a byte register. It is used for precise specification of frame format and desired baud rate. The parity bits, stop bits, baud rate selection and word length can be changed by writing the appropriate bits in LCR format which is shown in figure 2.

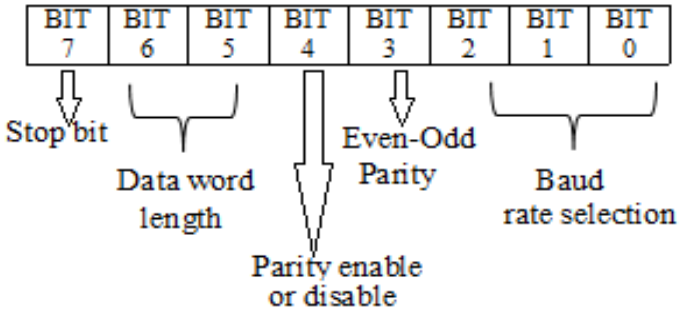

Fig.2. Format for Line Control Register

B. Baud Rate Generator (BRG):

The baud rate generator is programmable by three control bits Bit $_0$, Bit $_1$ and Bit $_2$ in LCR.

C. Transmitter:

The transmitter section accepts parallel data, makes the frame of the data and transmits the data in serial form on the Transmitter Output (TXOUT) terminal. Data is loaded from the inputs TXIN0-TXIN14 into the Transmitter FIFO by applying logic high on the WR (Write) input. If words less than 8 bits are used, only the least significant bits are transmitted. FIFO is a 16-byte register. When FIFO contains some data, it will send the signal to Transmitter Hold Register (THR), which is an 8-bit register. At a same time, if THR is empty it will send the signal to FIFO, which indicates that THR is ready to receive data from FIFO. If Transmitter Shift Register (TSR) is empty it will send the signal to THR and it indicates that TSR is ready to receive data from THR. TSR is a 15-bit register in which framing process occurs. Figure 3 shows the flow chart for Transmitter (Input to FIFO) and Figure 4 shows the flow chart for Transmitter (FIFO to output).

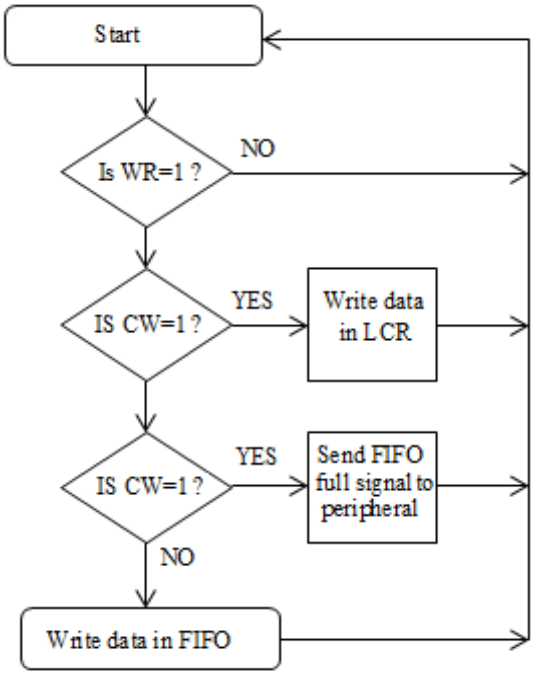

Fig. 3. Flow chart for Transmitter (Input to FIFO)

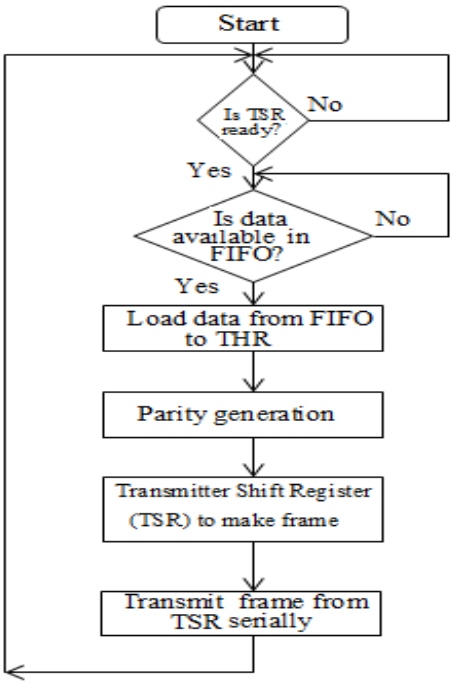

Fig. 4. Flow chart for Transmitter (FIFO to Output)

D. *Receiver:*

The transmitted data from the TXOUT pin is available on the RXIN pin. The received data is applied to the sampling logic block. Figure 5 shows the flow chart for receiver (Input to FIFO) and Figure 6 shows the flow chart for receiver (FIFO to output).

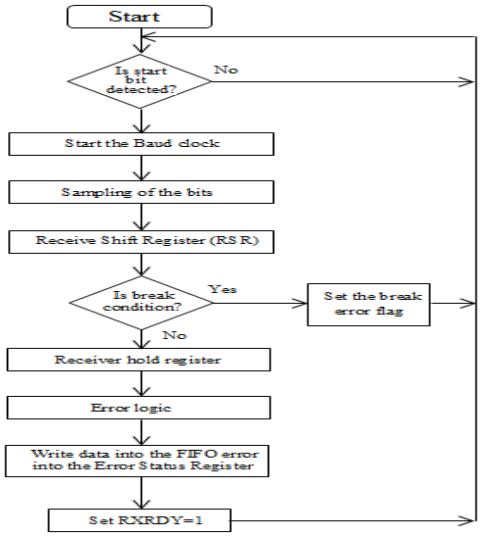

Fig.5. Flow chart for receiver (Input to FIFO)

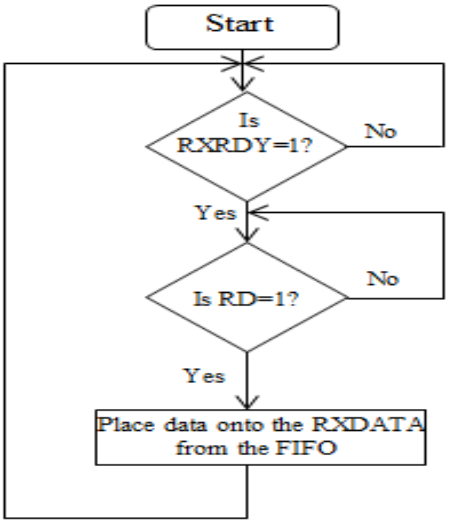

Fig. 6. Flow chart for receivr (FIFO to Output)

The receiver timing and control is used for synchronization of clock signal between transmitter and receiver. Initially the logic line is high whenever it goes low sampling and logic block will take 4 samples of that bit and if all four are same it indicates the start of a frame. After that remaining bits are sampled in the same way and all the bits are sent to Receiver Shift Register (RSR) one by one where the entire frame is stored. RSR is a 15 bit shift register. Now if the Receiver Hold Register (RHR) is empty it sends signal to RSR so that only the data bits from RSR goes to RHR which is an 8 bit register. The remaining bits in the RSR are used by the error logic block. Now if receiver FIFO is empty, it sends the signal to RHR so that the data bits goes to FIFO. When RD signal is asserted, the data is available in parallel form on the RXOUT (0-14) pins. The error logic block handles 4 types of errors: Parity error, Frame error, Overrun error and break error. If the received parity does not match with the parity generated from data bits, then PL bit will be set which indicates that parity error has occurred. If receiver fails to detect correct stop bit or when 4 samples do not match, then frame error occurs and SL bit is set. If the receiver FIFO is full and other data arrives at the RHR, overrun error occurs and OL bit is set. If the RXIN pin is held low for long time than the frame time, then there is a break in received data and break error occurs and BL bit is set.

E. Hamming Encoder:

The process of encoding is shown below:
1. Decide on the number of bits in the code word
2. Determine the bit positions of the check bits
3. Determine which parity bits check which positions
4. Calculate the values of the parity bits

The bit positions covered by each parity bit can be calculated by writing each bit position as a sum of the powers of 2:
1 = 1; 2 = 2; 3 = 1 + 2; 4 = 4; 5 = 1 + 4; 6 = 2 + 4; 7 = 1 + 2 + 4; 8 = 8; 9 = 1 + 8; 10 = 2 + 8; 11 = 3 + 8; 12 = 4 + 8
Thus, Check bit 1 governs positions 1, 3, 5, 7, 9: Value = 1 (0 0 0 1)
Check bit 2 governs positions 2, 3, 6, 7, 10: Value = 0 (0 0 0 0)
Check bit 4 governs positions 4, 5, 6, 7, 12 = 0 (0 0 0 0)
Check bit 8 governs positions 8, 9, 10, 11, 12 = 0 (1 0 1 0)
Thus, the complete code word is 1000 0000 1010.
The process of decoding is shown below:
A 1-bit error in this code word can be corrected as follows:

First, calculate the parity bits. If all are correct, there is either no error, or errors in more than 1 bit. However, we are limiting ourselves to single-bit errors here. If the parity bits show an error, add up all the erroneous parity bits, counting 1 for bit 1, 2 for bit 2, 4 for bit 4 and 8 for bit 8. The result gives the position of the erroneous bit, which can be complemented to give the correction.

For example, assume the code word is corrupted to give 1010 0000 1010.
Check parity:
Bit 1: $B_1 \_ B_3 \_ B_5 \_ B_7 \_ B_9 = 1 \_ 1 \_ 0 \_ 0 \_ 1 = 1$
Bit 2: $B_2 \_ B_3 \_ B_6 \_ B_7 \_ B_{10} = 0 \_ 1 \_ 0 \_ 0 \_ 0 = 1$
Bit 4: $B_4 \_ B_5 \_ B_6 \_ B_7 \_ B_{12} = 0 \_ 0 \_ 0 \_ 0 \_ 0 = 0$
Bit 8: $B_8 \_ B_9 \_ B_{10} \_ B_{11} \_ B_{12} = 0 \_ 1 \_ 0 \_ 1 \_ 0 = 0$
Thus the error is in bit position 1 + 2 = 3, and bit 3 can be inverted to give the correct code word:1000 0000 1010

III. SIMULATION AND RESULTS

The designed UART is synthesized using Xilinx project navigator. The resource utilization of entire transmitter & receiver module is as shown in Table I.

Table I Resource utilization of entire transmitter and receiver

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**% Utilization**
No. of Slices	136	4656	2
No. of slice FFs	85	9312	0
No. of 4 i/p LUTs	258	9312	2
No. of IOBs	30	232	12
No. of GCLKs	1	24	4

The synthesis diagram of the top module of UART is shown in the figure 7.

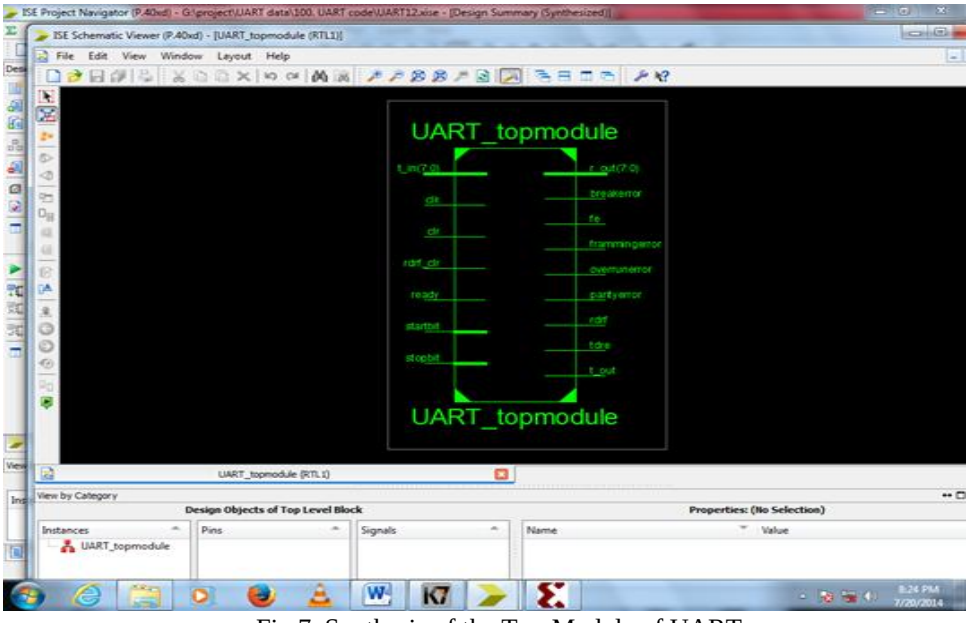

Fig.7. Synthesis of the Top Module of UART

The technology schematic report of the top module of UART is shown in the figure 8.

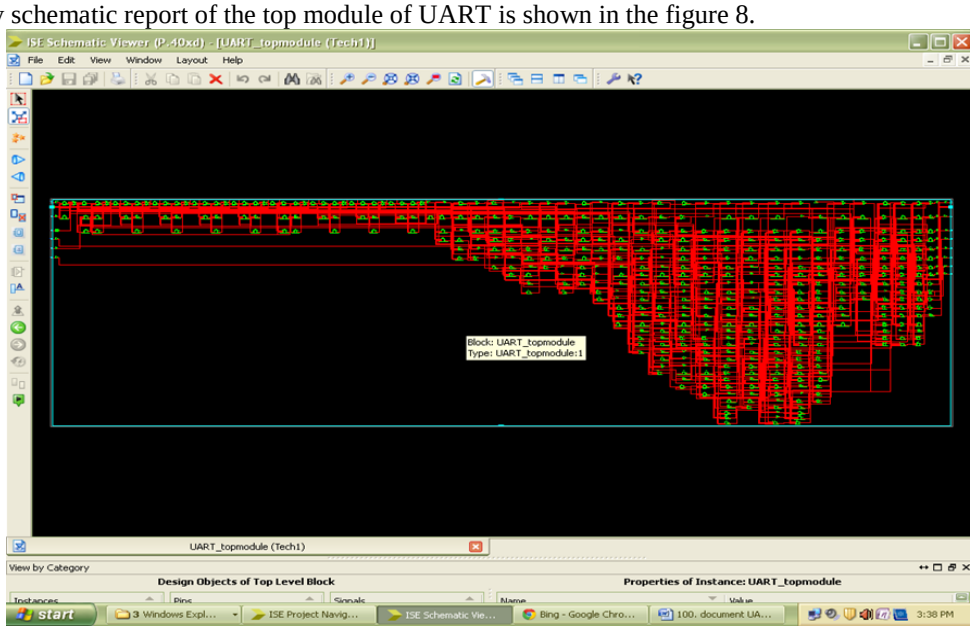

Fig.8. Technology Schematic of the Top Module of UART

The simulation results for the top module are shown in the figure 9.

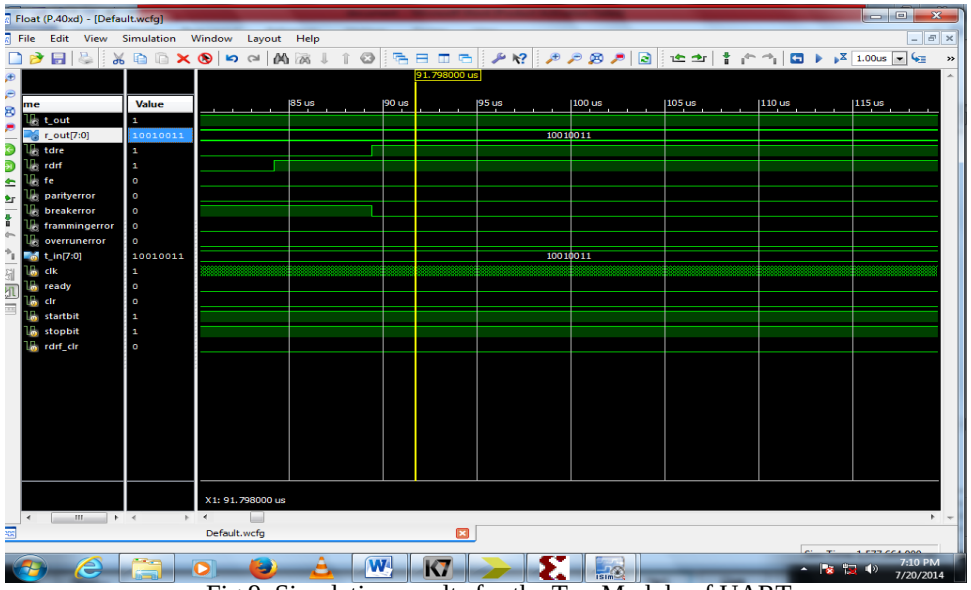

Fig.9. Simulation results for the Top Module of UART

The synthesis report generated is shown in the figure 10.

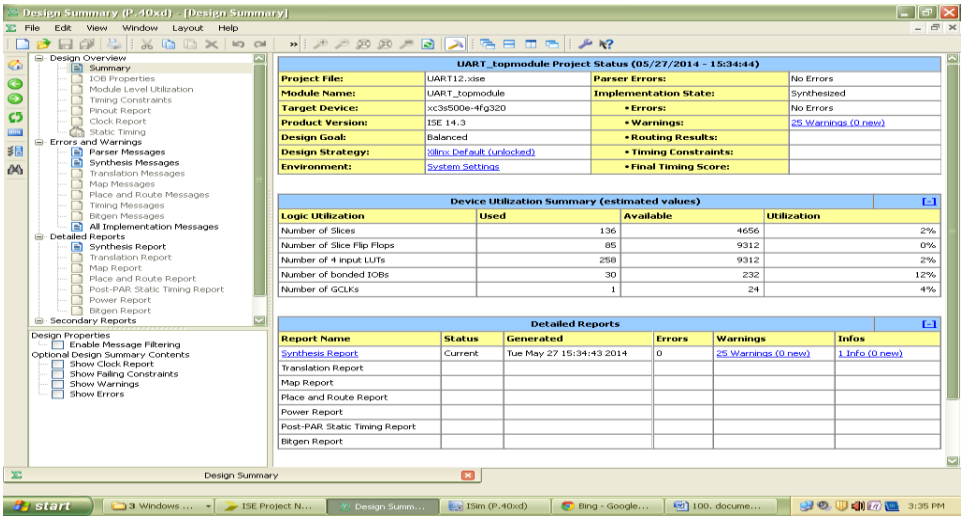

Fig.10. Synthesis report for the Top Module of UART

IV. CONCLUSION

Universal Asynchronous Receiver Transmitter (UART) is a broadly used serial data transmission protocol which supports full-duplex serial communication. It includes advantages, such as reliable performance, simple resources, easy to operate and realize strong anti-jamming capability, and much more. The architecture of UART can detect various errors such as parity error, stop error, overflow error and break error. This paper describes the architecture of UART that support various data word length, parity selection and different baud rates for serial transmission of data. In the propose architecture we are implement hamming code for error correction. Working principle of this UART has been tested using ISE simulator, which can be implemented on FPGA. Additionally, we can detect the different types of errors occurred during communication and hence correct them.

REFERENCES

[1] Mohd Yamani Idna Idris, Mashkuri Yaacob and Zaidi Razak, "A VHDL Implementation of UART Design with BIST Capability", *In the proceedings of Malaysian Journal of Computer Science*, June 2006, Vol. 19(1), pp. 73-86.

[2] Himanshu Patel; Sanjay Trivedi; R. Neelkanthan; V. R.Gujraty; , "A Robust UART Architecture Based on Recursive Running Sum Filter for Better Noise Performance," VLSI Design, 2007. Held jointly with 6thInternational Conference on Embedded Systems., 20th International Conference on Embedded Systems, vol., no., pp.819-823, Jan. 2007.

[3] Idris, M.Y.I.; Yaacob, M.; , "A VHDL implementation of BIST technique in UART design," TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region , vol.4, no., pp.1450- 1454 Vol.4, 15-17 Oct. 2003.

[4] Elmenreich W.;Delvai, M.; , "Time-triggered communication with UARTs," Factory Communication Systems, 2002. 4th IEEE International Workshop on , vol., no., pp. 97 - 104, 2002.

[5] Mahat N.F, "Design of a 9-bit UART module based on Verilog HDL", *in the proceedings of 10th IEEE International Conference on Semiconductor Electronics (ICSE), (*19-21), September. 2012, pp. 570-573.

[6] Yongcheng Wang; Kefei Song; , "A new approach to realize UART," Electronic and Mechanical Engineering and Information Technology
(EMEIT), 2011 International Conference on , vol.5, no., pp.2749- 2752, 12-14 Aug. 2011.M. Young, The Technical Writer's Handbook. Mill
Valley, CA: University Science, 1989.

[7] Fang Yi-yuan; Chen Xue-jun; , "Design and Simulation of UART Serial Communication Module Based on VHDL," Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on , vol., no., pp.1-4, (28-29) May 2011.

[8] Norhuzaimin, J.; Maimun, H.H.; , "The design of high speed UART," Applied Electromagnetics, 2005. APACE 2005.

[9] C. He, Y. Xia, and L. Wang, "A universal asynchronous receiver transmitter design", *International Conference on Electronics Comm. and Control (ICECC 2011)*, Ningbo, China, September. 2011.

[10] Liakot Ali, RoslinaSidek, IshakAris, AlauddinMohd. Ali, and Bam- bangSunaryo, "Design of a micro-UART for SOC application", Computer and Electrical Engineereing, vol 30, pp.257-268.

[11] Chun-zhi, He; Yin-shui, Xia; Lun-yao, Wang; , "A universal asynchronous receiver transmitter design," International Conference on Electronics, Communications and Control (ICECC), 2011, vol., no., pp.691-694, 9-11 Sept. 2011.

